Amendments to Claims

Listing of the claims

1. (currently amended) A method for executing database queries including identifying redundant conditions, comprising the steps of:

(a) identifying a first set of conditions corresponding to a selected step for executing a query;

(b) identifying a second set of conditions corresponding to one or more steps for executing the query that feed the selected step;

(c) for each condition in the first set, checking whether the condition is mathematically redundant, including redundant without being equivalent, with

the ~~other~~ conditions in the union of

~~the~~ any other conditions corresponding to the selected step and

the conditions in the second set,

(d) including each condition in the first set that is redundant as checked in step (c) in a third set; ~~and~~

(e) if there is only one condition in the third set after performing step (d), storing an identifier of the one condition;

(f) calculatin g a cardinality of the selected step without considering any condition with a stored identifier; and

(g) executing the query based at least in part on the cardinality.

2. (cancelled)

3. (currently amended) The method of claim 1, further comprising the steps of:

(~~f~~d1) if there are a plurality of conditions in the third set after performing step (d), performing steps (~~g~~d2) and (~~h~~d3);

(~~g~~d2) sorting the conditions in the third set to identify an initial condition; and

(~~h~~d3) storing an identifier of the initial condition.

4. (currently amended) The method of claim 1, wherein step (e) includes ceasing storage of identifiers of additional conditions and the union in step (c) does not include conditions with stored identifiers, and further comprising the steps of:

(~~f~~e1) if there are no conditions in the third set after performing step (d), ceasing storage of identifiers of additional conditions;

(~~g~~e2) if there are a plurality of conditions in the third set after performing step (d), performing steps (~~h~~e3) - (~~k~~e6);

(~~h~~e3) sorting the conditions in the third set to identify an initial condition;

(~~i~~e4) storing an identifier of the initial condition;

(~~j~~e5) modifying the first set of conditions to include only the conditions in the third set that are not the initial condition; and

(~~k~~e6) starting again at step (c).

5. (original) The method of claim 4, further comprising the step of performing a cardinality calculation without considering conditions with stored identifiers.

6. (original) The method of claim 1 where the one or more steps for executing the query that feed the selected step include all the steps that feed the selected step.

7. (original) The method of claim 1 further comprising the step of executing the query without applying conditions with stored identifiers.

8. (currently amended) The method of claim 1 further comprising the steps of:

(~~f~~e1) for each condition with a stored identifier that is a single table condition, performing steps (~~g~~e2)-(~~i~~e4);

(~~g~~e2) if the condition is an equality with an index of the single table, identifying that condition as CPU effective;

(~~h~~e3) if the condition is a range comparison of a value ordered index of the single table, identifying that condition as CPU effective;

(~~i~~e4) if the condition reduces the size of a temporary result in the execution of the query, identifying that condition as CPU effective; and

(~~j~~e5) executing the query without applying conditions with stored identifiers that are not identified as CPU effective.

9. (original) The method of claim 8 where identifying a condition as CPU effective comprises not identifying the condition as CPU redundant.

10. (currently amended) The method of claim 1 further comprising the steps of:

(~~f~~e1) for each condition with a stored identifier that is a join condition, performing step (~~g~~e2);

(~~g~~e2) if the condition is useful in co-locating the sources of the join, identifying that condition as CPU effective;

(~~h~~e3) executing the query without applying conditions with stored identifiers that are not identified as CPU effective.

11. (currently amended) A computer program, stored on a tangible storage medium, for executing database queries, the program including executable instructions that cause a computer to:

(a) identify a first set of conditions corresponding to a selected step for executing a query;

(b) identify a second set of conditions corresponding to one or more steps for executing the query that feed the selected step;

(c) for each condition in the first set, check whether the condition is mathematically redundant, including redundant without being equivalent, with

the ~~other~~ conditions in the union of

~~the~~ any other conditions corresponding to the selected step and

the conditions in the second set,

(d) include each condition in the first set that is redundant as checked in step (c) in a third set; ~~and~~

(e) if there is only one condition in the third set after performing step (d), store an identifier of the one condition;

(f) calculate a cardinality of the selected step without considering any condition with a stored identifier; and

(g) execute the query based at least in part on the cardinality.

12. (cancelled)

13. (currently amended) The program of claim 11, further including executable instructions that cause the computer to:

(~~f~~d1) if there are a plurality of conditions in the third set after performing step (d), perform steps (~~g~~d2) and (~~h~~d3);

(~~g~~d2) sort the conditions in the third set to identify an initial condition; and

(~~h~~d3) store an identifier of the initial condition.

14. (currently amended) The program of claim 11, wherein step (e) includes ceasing to store identifiers of additional conditions and the union in step (c) does not include conditions with stored identifiers, and further including executable instructions that cause the computer to:

(~~f~~e1) if there are no conditions in the third set after performing step (d), cease to store identifiers of additional conditions;

(~~g~~e2) if there are a plurality of conditions in the third set after performing step (d), perform steps (~~h~~e3) and (~~i~~e4);

(~~h~~e3) sort the conditions in the third set to identify an initial condition;

(~~i~~e4) store an identifier of the initial condition;

(~~j~~e5) modify the first set of conditions to include only the conditions in the third set that are not the initial condition; and

(~~k~~e6) start again at step (c).

15. (original) The program of claim 14, further including executable instructions that cause the computer to perform a cardinality calculation without considering conditions with stored identifiers.

16. (original) The program of claim 11 where the one or more steps for executing the query that feed the selected step include all the steps that feed the selected step.

17. (original) The program of claim 11 further including executable instructions that cause the computer to execute the query without applying conditions with stored identifiers.

18. (currently amended) The program of claim 11 further including executable instructions that cause the computer to:

(~~f~~e1) for each condition with a stored identifier that is a single table condition, perform steps (~~g~~e2)-(~~i~~e4);

(~~g~~e2) if the condition is an equality with an index of the single table, identify that condition as CPU effective;

(~~h~~e3) if the condition is a range comparison of a value ordered index of the single table, identify that condition as CPU effective;

(~~i~~e4) if the condition reduces the size of a temporary result in the execution of the query, identify that condition as CPU effective; and

(~~j~~e5) execute the query without applying conditions with stored identifiers that are not identified as CPU effective.

19. (original) The program of claim 18 where identifying a condition as CPU effective comprises not identifying the condition as CPU redundant.

20. (currently amended) The program of claim 11 further including executable instructions that cause the computer to:

(~~f~~e1) for each condition with a stored identifier that is a join condition, perform step (~~g~~e2);

(~~g~~e2) if the condition is useful in co-locating the sources of the join, identify that condition as CPU effective;

(~~he~~3) execute the query without applying conditions with stored identifiers that are not identified as CPU effective.

21. (currently amended) A database system for executing database queries, comprising:

one or more nodes;

a plurality of CPUs, each of the one or more nodes providing access to one or more CPUs;

a plurality of virtual processes, each of the one or more CPUs providing access to one or more virtual processes;

each virtual process configured to manage data, including rows organized in tables, stored in one of a plurality of data-storage facilities; and

an optimizer configured to:

(a) identify a first set of conditions corresponding to a selected step for executing a query;

(b) identify a second set of conditions corresponding to one or more steps for executing the query that feed the selected step;

(c) for each condition in the first set, check whether the condition is mathematically redundant, including redundant without being equivalent, with

the ~~other~~ conditions in the union of

~~the~~ any other conditions corresponding to the selected step and

the conditions in the second set,

(d) include each condition in the first set that is redundant as checked in step (c) in a third set; ~~and~~

(e) if there is only one condition in the third set after performing step (d), store an identifier of the one condition;

(f) calculate a cardinality of the selected step without considering any condition with a stored identifier; and

(g) execute the query based at least in part on the cardinality.

22. (cancelled)

23. (currently amended) The database system of claim 21, wherein the optimizer is further configured to:

(~~f~~d1) if there are a plurality of conditions in the third set after performing step (d), perform steps (~~g~~d2) and (~~h~~d3);

(~~g~~d2) sort the conditions in the third set to identify an initial condition; and

(~~h~~d3) store an identifier of the initial condition.

24. (currently amended) The database system of claim 21, wherein step (e) includes ceasing to store identifiers of additional conditions and the union in step (c) does not include conditions with stored identifiers, and wherein the optimizer is further configured to:

(~~f~~e1) if there are no conditions in the third set after performing step (d), cease to store identifiers of additional conditions;

(~~g~~e2) if there are a plurality of conditions in the third set after performing step (d), perform steps (~~h~~e3) and (~~i~~e4);

(~~h~~e3) sort the conditions in the third set to identify an initial condition;

(~~i~~e4) store an identifier of the initial condition;

(~~j~~e5) modify the first set of conditions to include only the conditions in the third set that are not the initial condition; and

(~~k~~e6) start again at step (c).

25. (original) The database system of claim 24, wherein the optimizer is further configured to perform a cardinality calculation without considering conditions with stored identifiers.

26. (original) The database system of claim 21 where the one or more steps for executing the query that feed the selected step include all the steps that feed the selected step.

27. (original) The database system of claim 21 wherein the optimizer is further configured to execute the query without applying conditions with stored identifiers.

28. (currently amended) The database system of claim 21 wherein the optimizer is further configured to:

(~~f~~e1) for each condition with a stored identifier that is a single table condition, perform steps (~~g~~e2)-(~~i~~e4);

(~~g~~e2) if the condition is an equality with an index of the single table, identify that condition as CPU effective;

(~~h~~e3) if the condition is a range comparison of a value ordered index of the single table, identify that condition as CPU effective;

(~~i~~e4) if the condition reduces the size of a temporary result in the execution of the query, identify that condition as CPU effective; and

(~~j~~e5) execute the query without applying conditions with stored identifiers that are not identified as CPU effective.

29. (original) The database system of claim 28 where identifying a condition as CPU effective comprises not identifying the condition as CPU redundant.

30. (currently amended) The database system of claim 21 wherein the optimizer is further configured to:

(~~f~~e1) for each condition with a stored identifier that is a join condition, perform step (~~g~~e2);

(~~g~~e2) if the condition is useful in co-locating the sources of the join, identify that condition as CPU effective;

(~~h~~e3) execute the query without applying conditions with stored identifiers that are not identified as CPU effective.